1. A system-on-a-chip (SOC) having a plurality of agents which share a memory, the SOC comprising:

a first bus interface coupled to a first one of the plurality of agents for interfacing it to the memory;

a second bus interface coupled to a second one of the plurality of agents for interfacing it to the memory;

wherein said first and second bus interfaces comprise:

request logic for submitting requests;

snoop logic, for responding to snoops; and

response logic, for receiving requested data; and

a memory controller coupled to the memory, said memory controller receiving requests from said first and second agents, and for globally ordering said requests;

wherein responses to said requests are latency independent.

2. The system-on-a-chip as recited in claim 1 wherein the plurality of agents comprise microprocessor cores.

3. The system-on-a-chip as recited in claim 2 wherein each of said microprocessor cores include a cache.

4. The system-on-a-chip as recited in claim 1 wherein said first and second bus interfaces and said memory controller are coupled to a common bus.

5. The system-on-a-chip as recited in claim 1 wherein said first and second bus interfaces and said memory controller are coupled to disparate buses.

6. The system-on-a-chip as recited in claim 1 wherein said first and second bus interfaces provide coherency between caches within the first and second plurality of agents and the memory.

7. The system-on-a-chip as recited in claim 1 wherein said request logic submits read and write requests to the memory.

8. The system-on-a-chip as recited in claim 1 wherein said snoop logic monitors read and write requests to determine whether another one of the plurality of agents requests data that is within its agent.

9. The system-on-a-chip as recited in claim 8 wherein if said snoop logic determines that another one of the plurality of agents requests data that is within its agent, said snoop logic informs said memory controller.

10. The system-on-a-chip as recited in claim 1 wherein said response logic associates received requested data with requests from its agent.

11. A microprocessor based system comprising:

a first microprocessor having a cache for caching data from a memory, said first microprocessor having a bus interface that implements a plurality of bus phases for a memory transaction;

a second microprocessor having a cache for caching data from said memory, said second microprocessor having a bus interface that implements a plurality of bus phases for a memory transaction;

a memory controller, coupled to said memory; and

a global arbiter, coupled to said bus interface of said first microprocessor, to said bus interface of said second microprocessor, and to said memory controller, said global arbiter for receiving requests for memory transactions from said first and second microprocessors, and for globally ordering said requests during request phases of said plurality of bus phases, and for initiating snoop phases to said bus interfaces to insure coherency, said snoop phases conforming to the globally ordering;

wherein said snoop phases are latency independent with respect to said request phases.

12. The microprocessor based system as recited in claim 11 wherein said plurality of bus phases implemented by said bus interfaces comprise:

a request phase, where memory requests are produced by said bus interfaces;

a snoop phase, where requests produced by said bus interfaces are observed and responded to; and

a response phase, where data associated with a request is received.

13. The microprocessor based system as recited in claim 12 wherein said memory requests comprise:

a request to share;

a request to own; and

a write back.

14. The microprocessor based system as recited in claim 13 wherein said request to share results from a read miss in a cache.

15. The microprocessor based system as recited in claim 13 wherein said request to own results from a write miss in a cache.

16. The microprocessor based system as recited in claim 13 wherein said write back results from a snoop to a modified cache line.

17. The microprocessor based system as recited in claim 11 wherein said first microprocessor's bus interface comprises:

request logic for submitting requests to said memory controller;

snoop logic, for observing requests by said second microprocessor; and

response logic, for receiving responses to requests submitted by said request logic.

18. The microprocessor based system as recited in claim 11 wherein said first microprocessor's bus interface couples said first microprocessor to a first bus.

19. The microprocessor based system as recited in claim 18 wherein said second microprocessor's bus interface couples said second microprocessor to a second bus.

20. The microprocessor based system as recited in claim 19 wherein said first bus and said second bus are the same.

21. The microprocessor based system as recited in claim 19 wherein said first bus and said second bus are different.

22. The microprocessor based system as recited in claim 19 wherein said first bus and said second bus have different timing latencies.

23. The microprocessor based system as recited in claim 11 wherein said memory controller comprises:

a bus interface for coupling to said global arbiter.

24. The microprocessor based system as recited in claim 11 wherein said global arbiter comprises a bus interface for coupling to each of said first and second microprocessors and said memory controller.

25. The microprocessor based system as recited in claim 11 wherein said global arbiter comprises a first bus interface for coupling to said first microprocessor, a second bus interface for coupling to said second microprocessor, and a third bus interface for coupling to said memory controller.

26. The microprocessor based system as recited in claim 11 wherein said global arbiter comprises:

a plurality of request queues;

a plurality of snoop queues; and

a plurality of response queues;

wherein each of said queues stores a plurality of requests, snoops, and responses, respectively.

27. The microprocessor based system as recited in claim 11 wherein said global arbiter comprises:

ordering and arbitration logic for receiving a plurality of requests from said first and second microprocessors, and for ordering said plurality of requests into a global order.

28. The microprocessor based system as recited in claim 27 wherein said ordering and arbitration logic initiates snoops according to said global order.

29. The microprocessor based system as recited in claim 28 wherein said snoops have differing latencies depending on bus characteristics of busses coupling said global arbiter to said first and second microprocessors.

30. A global arbiter for use in processor based system to insure coherency between a memory and a plurality of agents, the global arbiter comprising:

request logic, for receiving requests from each of the plurality of agents;

snoop logic, for initiating snoops to the plurality of agents; and

ordering logic, coupled to said request logic and said snoop logic, for establishing a global order for said requests, and for insuring that said initiated snoops conform to said global order;

wherein said initiated snoops are latency independent with respect to said requests.

31. The global arbiter as recited in claim 30 wherein said plurality of agents comprise:

processor cores; and

I/O devices.

32. The global arbiter as recited in claim 31 wherein said processor cores each have a cache.

33. The global arbiter as recited in claim 30 wherein said request logic comprises a plurality of queues for receiving said requests from the plurality of agents and for presenting said requests to said ordering logic.

34. The global arbiter as recited in claim 30 wherein said snoop logic comprises a plurality of queues for storing snoops conforming to said global order.

35. The global arbiter as recited in claim 30 wherein said ordering logic causes said snoop logic to initiate snoops according to said global order.

36. The global arbiter as recited in claim 30 wherein said global order is established according to a first-in first-out rule.

37. The global arbiter as recited in claim 30 wherein said ordering logic insures that responses to said requests are performed according to said global order.

38. The global arbiter as recited in claim 30 further comprising:

response logic for insuring that responses to said requests are performed according to said global order.

39. The global arbiter as recited in claim 30 further comprising:

a plurality of bus interfaces for coupling said global arbiter to the plurality of agents.

40. The global arbiter as recited in claim 39 wherein at least one of said plurality of bus interfaces couples to a bus that is different than the others.

41. The global arbiter as recited in claim 39 wherein at least one of said plurality of bus interfaces couples to a bus that supports a plurality of virtual channels.

42. The global arbiter as recited in claim 41 wherein said at least one of said plurality of bus interfaces comprises a bus interface to PCI-Express.

43. A multiphase protocol for insuring coherency between agents that share a memory through disparate fabrics, the protocol comprising:

a request phase, where memory requests are presented to a global arbiter, said global arbiter ordering said memory requests into a global order;

a snoop phase, where snoops are presented to agents coupled to the disparate fabrics according to said global order; and

a response phase, where responses to said memory requests are provided according to said global order upon completion of their associated snoops.

44. The multiphase protocol as recited in claim 43 wherein the agents comprise a plurality of processor cores.

45. The multiphase protocol as recited in claim 44 wherein each of the agents further comprise a cache.

46. The multiphase protocol as recited in claim 43 wherein the disparate fabrics comprise a plurality of interfaces.

47. The multiphase protocol as recited in claim 46 wherein at least one of said plurality of interfaces comprises a bus.

48. The multiphase protocol as recited in claim 43 wherein at least one of said plurality of interfaces comprises a serial fabric.

49. The multiphase protocol as recited in claim 48 wherein said serial fabric comprises PCI-Express.

50. The multiphase protocol as recited in claim 43 wherein said request phase, said snoop phase, and said response phase pertains to each of said memory requests.

51. The multiphase protocol as recited in claim 50 wherein said request, snoop, and response phases for one of said memory requests may overlap with said request, snoop, and response phases for another one of said memory requests.

52. The multiphase protocol as recited in claim 43 wherein said memory requests comprise:

memory reads; and

memory writes.

53. A method for providing latency independent coherence among a plurality of agents that share a memory, the method comprising:

establishing three phases for memory requests comprising:

a request phase;

a snoop phase; and

a response phase;

when multiple memory requests are outstanding, establishing a global order for the requests, each of the requests submitted during the request phase;

entering a snoop phase for each of the requests, following its request phase, the snoop phase querying the plurality of agents to determine whether they contain data pertaining to the requests; and

entering a response phase for each request after the plurality of agents have responded to the snoop phase for the request, the response phase providing data pertaining to the request to its requesting agent.

54. The method as recited in claim 53 wherein the request phase comprises submitting a request from an agent to a global arbiter.

55. The method as recited in claim 54 wherein the request phase further comprises receiving the request by the global arbiter, and ordering the request according to the global order.

56. The method as recited in claim 53 wherein the snoop phase comprises communicating requests to the agents that share the memory according to the global order.

57. The method as recited in claim 56 wherein the snoop phase further comprises awaiting a snoop response from the agents before proceeding to the response phase.

58. The method as recited in claim 53 wherein the response phase comprises providing data associated with a request to the agent that generated the request, according to the global order.

59. The method as recited in claim 53 wherein the global order causes response phases associated with each request to be completed, in order, without regard to latencies between memory requests.

60. The method as recited in claim 53 wherein the global order causes response phases associated with each request to be completed, in order, without regard to latencies between a global arbiter and its agents.

61. A computer program product for use with a computing device, the computer program product comprising:

a computer usable medium, having computer readable program code embodied in said medium, for causing a system-on-a-chip, having processor cores each having a cache and sharing a memory, to be described, said computer readable program code comprising:

first program code for providing a global arbiter, coupled to each of the processor cores; and

second program code for providing a bus interface for each of the processor cores to couple the processor cores to the global arbiter;

third program code for providing a bus interface to couple the global arbiter to the memory;

wherein said global arbiter receives memory requests from each of the processor cores, establishes a global order for the requests, and initiates a snoop phase for each of the requests according to the global order; and

wherein the processor cores respond to the snoop phase initiated by the global arbiter at different times.

62. The computer program code product as recited in claim 61 wherein the global arbiter comprises:

request logic, for receiving requests from the processor cores;

snoop logic, for communicating received requests from the global arbiter to the processor cores; and

ordering logic, for establishing a global order for received requests, and for causing said snoop logic to communicate the received requests to the processor cores according to the global order.

63. The computer program code product as recited in claim 62 wherein the global arbiter further comprises:

response logic, for causing responses to the received requests to be provided upon completion of their associated snoop.

64. The computer program code product as recited in claim 62 wherein the global arbiter further comprises:

response logic, for causing responses to the received requests to be provided to the processor cores according to the global order.

65. A computer data signal embodied in a transmission medium comprising:

computer-readable program code for providing a latency independent coherence protocol, said program code comprising:

first program code for providing a request phase for a memory request to be transmitted from an agent to a global arbiter;

second program code for providing a snoop phase where the memory request is ordered by said global arbiter according to a global order, and transmitted to other agents according to said global order; and

third program code for providing a response phase where the response to said memory request is provided to said agent, upon completion of said snoop phase;

wherein by establishing a global order for memory requests, a memory is shared by a plurality of agents that are connected to said global arbiter by fabrics having different latencies.

66. The computer data signal as recited in claim 65 wherein said global arbiter comprises:

a plurality of bus interfaces, one for each of said fabrics;

request logic, coupled to said plurality of bus interfaces, for receiving said memory request;

ordering logic, coupled to said request logic, for receiving said memory request from said request logic, and for establishing said global order upon receipt; and

snoop logic, coupled to said ordering logic, for submitting snoop requests to said other agents according to said global order.

67. The computer data signal as recited in claim 66 further comprising:

response logic, for providing a response to said memory request when said snoop requests are completed.

68. The computer data signal as recited in claim 67 wherein said snoop requests are completed when each of said other agents have responded to said global arbiter.